Databinding: Passing Data Between Controller Classes and JSP Pages

Table of contents

1. Introduction

Your web application's presentation layer (JSP pages) and processing layer (Controller classes) are bound together through the process of *databinding*. There are numerous databinding contexts available, including all of the JSP 2.0 implicit objects as well as other contexts peculiar to page flow apps.

Databinding syntax uses the JSP 2.0 Expression Language. For a quick summary of the JSP 2.0 EL see JSP 2.0: The New Deal, Part 1 (http://www.onjava.com/pub/a/onjava/2003/11/05/jsp.html) .

It is important to note that the databinding contexts form a two-way street between the presentation and processing layers of a web application. Databinding contexts not only pass data from the processing layer to the presentation layer, some contexts can pass data in the opposite direction from the presentation layer to the processing layer.

The one-way, read-only contexts are: requestScope, param, sessionScope, container, pageScope, bundle, pageInput, applicationScope

The two-way, read-write contexts are: pageFlow, sharedFlow, actionForm

2. Summary of Binding Contexts

Context Name	Object the Context References	Description
`actionForm`	The current form bean (the form bean named in the current `<netui:form>` tag)	Individual fields defined in the form bean can be referenced. Values are read-write.
`applicationScope`	an attribute map on the application object (a JSP 2.0 implicit object)	An implicit JSP object. Attributes of this object are available across the web app.
`bundle`	Properties defined in a message resource file	On a JSP page, declare the message resource file using the <netui-data:declareBundle> tag. `<netui-data:declareBundle name="myBundle" bundlePath="properties.bundle1"/>` Reference individual properties in the file using the `bundle` databinding context. `<netui:span value="${bundle.myBundle.message1}"/>`

`container`	The data set referenced in the current <netui-data:dataGrid> or <netui-data:repeater> tag	In the example below, the `container` context refers to the object named in the `<netui-data:dataGrid>`'s `dataSource`
		`<netui-data:dataGrid dataSource="pageScope.stocks" name="portfolio">`
		`<netui-data:rows>`
		`<netui-data:spanCell value="${container.item.symbol}"/>`
		`<netui-data:spanCell value="${container.item.price}"/>`
		`</netui-data:rows>`
		`</netui-data:dataGrid>`
		See below for detailed information on `container`'s sub-contexts.

`pageFlow`	The current Controller class	Public members of the current Controller class can be referenced and written to. Assume that the Controller class contains the following member String and associated get and set methods.
		`public String someText = "This is some text"`
		`public String getSomeText(){`
		`return someText;`
		`}`
		`public void setSomeText(String newText){`
		`someText = newText;`
		`}`
		Then that member can be referenced from any JSP page in the page flow.
		`<netui:span value="${pageFlow.someText}"/>`

`pageInput`	an attribute map on the current Forward object	Page input data can be added to the Forward object using the `addActionOutput(String, Object) method..`
		`Forward forward = new Forward("success");`
		`forward.addActionOutput("name", someNameData);`

		The data can be retrieved on the JSP page using the `pageInput` databinding context. `<netui:span value="${pageInput.name}"/>`
`pageScope`	an attribute map on the current JSP page (a JSP 2.0 implicit object)	An implicit JSP object. Attributes of this object are available on the current JSP page.
`requestScope`	an attribute map on the request object (a JSP 2.0 implicit object)	An implicit JSP object. Attributes of this object are available for the life of the current request.
`sessionScope`	an attribute map on the session object (a JSP 2.0 implicit object)	An implicit JSP object. Attributes of this object are available for the life of the user session.

`sharedFlow`	Attribute map of all of the shared flows in the web application	Provided that the shared flow is declared in the Controller file
		`@Jpf.Controller( sharedFlowRefs={ @Jpf.SharedFlowRef(name="mySharedFlo type=sharedFlows.SharedFlow.class) } ) public class Controller extends PageFlowController`
		You can access the members of the shared flow through the `sharedFlow` context.
		`<netui:span value="${sharedFlow.mySharedFlow.some`

`param`	Query parameters on the current JSP's URL.	Query parameters that are part of the URL can be referenced. Values are read only.

3. Individual Binding Contexts

3.1. actionForm

To bind to data in a form bean, you use the `actionForm` context. In the following example

a JSP's form fields are pre-populated by binding to the fields of a form bean. Assume that you navigate to the JSP page via the following action method. Notice that the `Forward` object is constructed with two parameters: the first parameter is the string "update", the second parameter is a form bean (DatabaseForm). By constructing the Forward object in this way, the `actionForm` context is automatically filled with the form bean values, values which can be retrieved by the navigated-to JSP page.

```
@Jpf.Action(
    forwards={
            @Jpf.Forward( name="update", path="updateItems.jsp" )
        }
)
public Forward updateItems(DatabaseForm form)
{
    ...
        // Read from the database
        // and apply the values to the form bean.
    form.applyValuesToForm(getCurrentRow());

        // Construct a Forward object using the form bean.
    return new Forward("update", form);
}
```

The action method above reads a record from a database using a Database control (the control code is omitted in the example) and populates the fields in the form bean with the record data before navigating to the JSP `updateItems.jsp`. The values are retrieved by the JSP page, using the `actionForm` context:

```
<netui:form action="submitUpdate">
    <netui:content value="${actionForm.itemnumber}"/>
    <netui:textBox dataSource="actionForm.itemname"/>
    ...
</netui:form>
```

When the <netui:form> is submitted (`submitUpdate`), you can then write the updated values back to the database:

```
@Jpf.Action(
    forwards={
            @Jpf.Forward( name="updated", path="getItems.jsp" )
        }
)
public Forward submitUpdate(DatabaseForm aDatabaseForm)
{
    // code that calls the database control and updates the database
    return new Forward("updated");
}
```

3.2. applicationScope

3.3. bundle

3.4. container

The `container` refers to the data set passed into a <netui-data> or <netui-data:repeater> tag. The `container` allows access to the individual rows and cells of the data set.

The `container` context has the following properties.

Property Name	Description
`container.item`	Refers to the current row of the data set.
`container.item.[field_name]`	Refers to an individual field of the current row.
`container.index`	Refers to the integer index of the current row.

The following example shows how to access the name, type breed, and weight properties in a data set passed to a data grid.

The context `container` refers to the data set `pageInput.petList`.

```
<netui-data:dataGrid dataSource="pageInput.petList" name="petGrid">
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.type}"/>
        <netui-data:spanCell value="${container.item.breed}"/>
        <netui-data:spanCell value="${container.item.weight}""/>
    </netui-data:rows>
</netui-data:dataGrid>
```

3.5. pageFlow

When you define a variable in your controller class, you can access this variable from any JSP page that is part of that page flow by providing access with a getter and setter method. For example, if the JPF file contains this code:

```
  private String firstName;

  public String getFirstName() {
      return firstName;
  }

  public void setFirstName(String aStr) {
      firstName = aStr;
  }
```

You can access this variable on a JSP, for instance, you can bind the value to a netui:label as is shown next:

```
<netui:label value="${pageFlow.firstName}" />
```

In the example the method getFirstName is used to provide read access to the variable. The

variable is read-write and can be changed, for instance in the controller's action method as is shown in the next example:

```
@Jpf.Action(
    forwards={
            @Jpf.Forward( name="success", path="nextPage.jsp" )
        }
)
public Forward labelAction(NameActionForm form)
{
    ...
    firstName = "Default Name";
    return new Forward( "success");
}
```

It can also be changed in a JSP by using a form, as is shown next:

```
  <netui:form action="submitAction">
      <netui:textBox dataSource="pageFlow.firstName"/>
      ...
      <netui:button value="submit"/>
  </netui:form>
```

You can also define and access variables that are part of an object created in the controller class. You can provide this access by again using getter and setter methods. For instance, if the JPF file contains the code:

```
public class dataFlowController extends PageFlowController
{
    ...
    public Names myNames = new Names();
    ...
    public class Names implements Serializable {
        public String firstName;
        public String lastName;

        public String getFirstName(){
            return firstName;
        }
        public String getLastName(){
            return lastName;
        }
    }
    ...
```

You can access the firstName property in a JSP like this:

```
<netui:label value="${pageFlow.myNames.firstName}" />
```

The comments in this section regarding access of page flow properties also apply to properties defined to the `sharedFlow` context.

3.6. pageInput

The `pageInput` context is filled with data by an action output in the Controller class.

```
Forward f = new Forward("next_page");
f.addActionOutput("message", sMessage);
```

The data can be retrieved on the JSP page as follows:

```
<netui:span value="${pageInput.message}"/>
```

3.7. requestScope

3.8. sessionScope

You can use the `sessionScope` context to read data from the session object (an implicit JSP object). Assuming that the session object contains some data in its attribute map:

```
getSession().setAttribute("myAttributeKey", myObject);
```

then, you can read the data onto a JSP page as follows:

```
<netui:content
value="${sessionScope.myAttributeKey.someProperty}"></netui:content>
```

3.9. sharedFlow

If you need an attribute to be available across all the page flows in a web application's user (browser) session, such as a user's unique session key, you can use shared flow. By convention, shared flows are located in WEB-INF/src/shared. (For details on creating a shared flow, see Shared Flow (../pageflow/sharedFlow.html) .) In addition to session-wide attributes, you can also use the shared flow to define fallback action methods and exception handlers.

If you define a variable in the shared flow class:

```
public class SharedFlow extends SharedFlowController
{
    public String defaultText = "Please Define";

    public String getDefaultText(){
        return defaultText;
    }
    ...
}
```

and your controller file references the shared flow:

```
@Jpf.Contoller(
    sharedFlowsRefs = {
        @Jpf.SharedFlowRef(name = "specificSharedFlow", type =
"SharedFlow.class")
```

```
    }
)
public class SomeController extends Controller
{
...
}
```

you can use the databinding context `sharedFlow` in a JSP file:

```
<netui:textBox dataSource="actionForm.firstName"
defaultValue="${sharedFlow.specificSharedFlow.defaultText}"/>
```

Notice that the <netui:textBox> tag in the example binds to a form bean to post data and binds to sharedFlow's defaultText to receive its default value.

SharedFlow properties can be accessed from a JSP using the scope sharedFlow, for example,

```
dataSource="sharedFlow.specificSharedFlow.someProperty"
```

Actions declared in a SharedFlow are accessed by specifying a fully qualified action

```
action="specificSharedFlow.someAction"
```

where specificSharedFlow was the name given to the sharedFlow in the JPF annotation `Jpf.SharedFlowRef`.

3.10. param

4. Related Topics

For more information on the JSP 2.0 Expression Language, see JSP 2.0: The New Deal, Part 1 (http://www.onjava.com/pub/a/onjava/2003/11/05/jsp.html)